

PATENT APPLICATION TRANSMITTAL LETTER

(Large Entity)

Docket No.
INTL-0278-US

TO THE ASSISTANT COMMISSIONER FOR PATENTS

Transmitted herewith for filing under 35 U.S.C. 111 and 37 C.F.R. 1.53 is the patent application of:

John C. Dake and Paul E. Luse

For: **METHOD FOR MODELING HARDWARE USING SOFTWARE**

Enclosed are:


- ☒ Certificate of Mailing with Express Mail Mailing Label No. **EL515089232US**
- ☒ **Ten (10)** sheets of drawings.
- ☐ A certified copy of a _____ application.
- ☒ Declaration ☒ Signed. ☐ Unsigned.
- ☒ Power of Attorney
- ☐ Information Disclosure Statement
- ☐ Preliminary Amendment
- ☒ Other: **Recordation Form Cover Sheet, Assignment and check for \$40**

CLAIMS AS FILED

For	#Filed	#Allowed	#Extra	Rate	Fee
Total Claims	27	- 20 =	7	x \$18.00	\$126.00
Indep. Claims	6	- 3 =	3	x \$78.00	\$234.00
Multiple Dependent Claims (check if applicable) <input type="checkbox"/>					\$0.00
BASIC FEE					\$760.00
TOTAL FILING FEE					\$1,120.00

- ☒ A check in the amount of **\$1,120.00** to cover the filing fee is enclosed.
- ☒ The Commissioner is hereby authorized to charge and credit Deposit Account No. **20-1504** as described below. A duplicate copy of this sheet is enclosed.
 - ☐ Charge the amount of _____ as filing fee.
 - ☒ Credit any overpayment.
 - ☒ Charge any additional filing fees required under 37 C.F.R. 1.16 and 1.17.
 - ☐ Charge the issue fee set in 37 C.F.R. 1.18 at the mailing of the Notice of Allowance, pursuant to 37 C.F.R. 1.311(b).

Dated: **December 22, 1999**


 Timothy N. Trop, Reg. No. 28,994
 Trop, Pruner, Hu & Miles, P.C.
 8554 Katy Freeway, Suite 100
 Houston, Texas 77024
 Ph: (713) 468-8880
 Fax: (713) 468-8883

CC:

APPLICATION

FOR

UNITED STATES LETTERS PATENT

**TITLE: METHOD FOR MODELING HARDWARE USING
SOFTWARE**

INVENTORS: Steven C. Dake, Paul E. Luse

Express Mail No.: EL515089232US

Date: December 22, 1999

METHOD FOR MODELING HARDWARE USING SOFTWARE

Background

This invention relates to software programs and, more particularly, to application programming interfaces.

Software development is a robust industry. Following the development of
5 new hardware, software typically provides an interface to the hardware.

Before writing a software program, a developer may need to understand
the functional requirements of the software, the hardware upon which the
software may operate, and the operating environment. Particularly for complex
software, a logical organization of the software may also be beneficial to
10 reducing defects.

Some software programs are structured so that they work in a variety of
operating environments. Additionally, software that is well-written anticipates
new hardware to be supported by the software program. An application
programming interface, or API, is a tool for structuring such software
15 applications. As the name suggests, the API "interfaces" the core application
code with the rest of an operating environment.

APIs generally include specifications or protocols for their use. Programs
written according to the protocol may have a similar interface to the end-user,
for example. APIs also may include routines, libraries, or other tools which
20 minimize duplicity of effort for various developers who may perform similar
functions. Thus, APIs are tools that typically permit new development to support
new operating systems, to support new hardware, or to add new software
features to existing application programs.

Software which supports a number of hardware components may benefit from such interfaces. Further, software which supports hardware from more than a single vendor, each with a distinct hardware requirement, may interface to an API to simplify code development. For projects where the hardware is available only after the software is written, an API may also promote developing the software within the project time constraints.

Thus, there is a continuing need for an interface between a plurality of hardware devices and a software program to support the hardware devices such that the software program is simplified while continuing to support the hardware.

Summary

In general, in one embodiment of the invention, a method comprises defining a plurality of hardware devices as a plurality of objects, providing a plurality of tools to perform a plurality of operations on the plurality of objects, executing a software program to use the tools and responding to the plurality of operations by the plurality of hardware devices.

Other features and embodiments will become apparent from the following description, the drawings, and from the claims.

Brief Description of the Drawings

Figure 1 is a block diagram of an object model API according to one embodiment of the invention;

Figure 2 is a block diagram of an object of the object model API according to one embodiment of the invention;

Figure 3 is a block diagram of the configuration library of the object model API according to one embodiment of the invention;

Figure 4 is a block diagram of the RAID API according to one embodiment of the invention;

Figure 5 is a block diagram of RAID hardware in accordance with one embodiment of the invention;

5 Figure 6 is a block diagram of a plurality of RAID objects for the RAID hardware of Figure 5 in accordance with one embodiment of the invention;

Figure 7 is a block diagram of the controller object in accordance with one embodiment of the invention;

10 Figure 8 is a block diagram of the bus object in accordance with one embodiment of the invention;

Figure 9 is a block diagram of the disk object in accordance with one embodiment of the invention;

Figure 10 is a block diagram of the array object in accordance with one embodiment of the invention;

15 Figure 11 is a block diagram of the volume object in accordance with one embodiment of the invention; and

Figure 12 is a flow diagram of a bus scan operation using the RAID object API in accordance with one embodiment of the invention.

Detailed Description

20 In the prior art, a software program may be written to support one or more hardware devices. For example, the software program may include a graphical user interface which permits a user of the software program to control or monitor the hardware devices. The software program may initialize the hardware devices, as another example.

25 In supporting the hardware device, the software program may directly interact with the hardware device, or instead interact with an interface to the

hardware. The interface to the hardware device may itself be a second software program.

The software program may be written to support hardware devices from a particular vendor. In order to support hardware devices from a second vendor, the software program may be changed, the second vendor's hardware device or an interface to the second vendor's hardware device may be changed. For each distinct hardware type supported by the software program, the software program may grow proportionately.

An object model application programming interface (API) 10 may provide an alternative to the prior art. Turning to Figure 1, the object model API 10, according to one embodiment of the invention, may include a configuration library 20 and a plurality of objects 30. The object model API 10 is coupled between a software program 12 and a hardware device 16, a hardware device 17, and a hardware device 18.

The hardware device 16 may, for example, be supplied by a different vendor than the hardware device 17, etc. The hardware device 16 may itself include software, such as a software interface (not shown). The hardware device 16 may operate using one protocol, the hardware device 17 may operate using a second protocol, and so on.

The object model API 10 may reside between the software program 12 and the hardware devices 16-18. In one embodiment of the invention, the object model API 10 provides the capability to abstract the hardware devices 16-18 so that the software program 12 need not be written with the distinct requirements of the hardware devices 16-18 in mind. Instead, the software program 12 communicates with the objects 30 which model the hardware devices 16-18. In turn, the hardware devices 16-18 communicate with the

object model API 10, whether to take actions, provide information, or perform other operations on the hardware devices 16-18.

5 The object model API may be beneficial for some hardware/software models. The software program 12 may be written with less than full knowledge of the particular hardware involved. Instead, using the object model API 10, the software program 12 may be written once a "model" of the hardware devices 16-18 has been defined. The objects 30 represent this model.

10 Each hardware device 16-18 to be supported by the software program 12 communicates with the object model API 10. Where the hardware devices 16-18 include a software interface, the interface may communicate with the object model API 10. For example, the interface to the hardware devices 16-18 may respond to commands from the object model API 10 such that the commands are executed upon the hardware devices 16-18.

15 As part of the object model API 10, the configuration library 20 includes a set of tools which enable the software program 12 to communicate with the objects 30. The objects 30 model the hardware devices 16-18. Once the software program 12 knows how to retrieve and manipulate the objects 30, using the tools from the configuration library 20, the software program 12 may perform operations on the objects 30.

20 The objects 30, in essence, define the hardware devices 16-18 by modeling the operations (or methods), the attributes (or properties), and the actions (or events) of each hardware device 16-18. An operation, known as a method, may be invoked upon an object. The method is appropriate for the real-world hardware modeled by the object. One or more attributes, described
25 below as properties, may also be used to define an object. Actions, described

below as events, may model real-world occurrences by the hardware devices 16-18.

In one embodiment of the invention, each object encapsulates the operations of, the attributes of, and the actions to be taken upon the real-world hardware modeled by the object. For example, the hardware device 16 may be divided into components, each of which is modeled as a separate object 30. The characteristics for each component are identified, and become properties and methods of the object 30. Likewise, hardware devices 17 and 18 may have objects 30 for modeling their components.

In Figure 2, in one embodiment of the invention, the object 30 includes methods 32, properties 34 and events 36 for representing the characteristics of the hardware devices 16-18 modeled by the object 30. The methods 32 may include functions or actions which the hardware devices 16-18 modeled by the object 30 may perform.

For example, suppose the hardware device modeled is a staple gun. A staple gun may push staples into a piece of paper. Accordingly, an object modeling the staple gun may appropriately include a method, PushStaples.

The properties 34 may include attributes of the hardware device 16. For example, attributes of the staple gun may include the number and type of staples. An object modeling the staple gun may appropriately include two properties, NumberOfStaples and StapleType.

Further, the properties 34 may be defined as having different types. In the staple gun example, NumberOfStaples may be represented as an integer type, while StapleType may be a Boolean variable.

The events 36 may include those behaviors of the hardware devices 16-18 for which notification may be requested. For example, the staple gun may run

out of staples. Accordingly, an object modeling the staple gun may appropriately include an event, OutOfStaples.

5 The methods 32, the properties 34, and the events 36 of the object 30 thus model the behavior of the hardware devices 16-18. Once the hardware devices 16-18 are defined in terms of these features, the software program 12 may communicate with the object 30 instead of the hardware devices 16-18. In turn, the hardware devices 16-18 communicate with the object model API 10.

10 In one embodiment of the invention, the configuration library 20 may be used to communicate with the objects 30. In Figure 3, the tools of the configuration library 20 may include several functions. The functions may be used by the software program 12 to communicate with the objects 30.

15 The configuration library 20 includes an initialization function 52. The initialization function 52 initializes the configuration library 20. In one embodiment of the invention, the initialization function 52 is executed prior to any other function in the configuration library 20.

20 A second initialization function 54 allows the software program 12 to override any memory management functions of the environment. Some system management architectures provide unique memory management routines. Under such system management architectures, memory management routines may be replaced with other routines. The second initialization function 54 thus overrides the memory manager upon command.

25 The configuration library 20 further includes a set of functions for object creation and object discovery. An object locating function 58 provides the capability to locate an object of a specific type with a specific property value using a comparison operator as an input value. The object locating function 58 may be used to determine the components of the hardware device 16 by the

software program 12, for example. Closely associated with the object locating function 58, a subsequent object locating function 60 provides the capability to locate the next object based upon the parameters provided in the object locating function 58.

5 In one embodiment of the invention, an object creation function 56 allows the software program 12 to create an object for the hardware devices 16-18 which may not be physically located, e.g., virtual hardware features. The object creation function 56 essentially creates an internal storage area for the object 30. The internal storage area may be used for the properties 34 of the object 30
10 used to represent the virtual hardware feature, for example.

 Another tool in the configuration library 20, a method invocation function 62, allows the software program 12 to execute or invoke a method 32 within an object 30. The method 32 may include parameters which may be set prior to invoking the method 32 or may be returned upon executing the method 32. In
15 one embodiment of the invention, the parameters may include properties 34 of the object 30 for which the method 32 is invoked.

 In one embodiment of the invention, the configuration library 20 also provides functions for manipulating the properties 34 of the object 30. A get property function 64 enables the software program 12 to retrieve the property
20 34 of the object 30. Likewise, a set property function 66 allows the software program 12 to set the property 34 of the object 30.

 The configuration library 20 also provides functions which enable the software program 12 to be notified when an event to one of the hardware devices 16-18 occurs. A monitor event function 68, when enabled, allows the
25 software program 12 to be notified upon the occurrence of the particular event 36 of the object 30. Likewise, a monitor event off function 70 allows the

software program 12 to turn off monitoring of the particular event 36 of the object 30.

Thus, in one embodiment of the invention, the configuration library 20 of the object model API 10 provides the capability to invoke the methods 32, set and retrieve the properties 34, and monitor the events 36 for a given object 30. The software program 12 may utilize the functions in the configuration library 20 to communicate with the objects 30 which model the hardware devices 16-18.

The object model API 10 may be used as a front end to a variety of types of hardware devices 16-18. By distilling the hardware devices 16-18 into distinct components, an object model 30 for each component may then be created. Like the staple gun, described above, a variety of hardware devices may then be defined in terms of operations (methods), attributes (properties) and actions (events). These methods 32, properties 34, and events 36 are then encapsulated as objects 30. To support the hardware devices 16-18, a developer may then use the functions of the configuration library 20, described above, to perform operations upon the objects 30.

Because the software program 12 does not communicate directly with the hardware devices 16-18, the requirements of the hardware devices 16-18 are unimportant to the software program 12. Instead, the software program 12 communicates with the objects 30 using the tools of the configuration library 20. In this way, the software program 12 may support the hardware devices 16-18, even when the hardware devices 16-18 change. Further, the software program 12 may support future hardware additions.

Software which supports a redundant array of independent disks, or RAID, system, may benefit from the object model API 10. A RAID system is essentially a collection of disks, connected by one or more buses, and employing one or

more controllers. RAID systems may employ one or more disk drives, typically to provide fault tolerance and to enhance performance. In addition to the disk drives themselves, the RAID system may include a plurality of controllers and buses. Further, the RAID system may be organized into physical representations of disks, called arrays, or logical representations of disks, called volumes. The RAID hardware, as well as its supporting software, may therefore be complex.

Turning to Figure 4, a particular implementation of the object model API 10, shown as a RAID object API 10a, includes the configuration library 20 and RAID objects 30a, in accordance with one embodiment of the invention. The RAID API 10a interfaces with a RAID software program 12a and a RAID hardware device 16a, a RAID hardware device 17a, and a RAID hardware device 18a.

As in Figure 1, the RAID hardware devices 16a-18a may include RAID hardware device 16a, from one vendor, RAID hardware 17a, possibly from a second vendor, and RAID hardware 18a, possibly from a third vendor. Each RAID hardware device 16a-18a may include a software interface (not shown). The RAID hardware devices 16a-18a are responsive to directives from the RAID API 10a. Likewise, the RAID software 12a communicates with the objects 30a modeling the RAID hardware devices 16a-18a using the configuration library 20.

As with the object model API 10, with the RAID API 10a, RAID objects 30a may be defined according to the RAID hardware devices 16a-18a. The RAID hardware devices 16a-18a may also be described in terms of physical groupings of disks into a unified storage medium, known as an array. Alternatively or additionally, the RAID hardware devices 16a-18a may be described in terms of logical groupings, or volumes.

5 The RAID hardware 16a-18a may be distilled into distinct components, which are then modeled as objects 30a. In Figure 5, a system 14 according to one embodiment of the invention may include a processor 90 and a memory 92 are coupled to a processor bus 94. A secondary bus 98 may be coupled to the processor bus 94 via a bridge 96.

10 The RAID hardware devices 17a extend from the secondary bus 98, in one embodiment of the invention. Controllers 100A and 100B are coupled to the secondary bus 98. From the controllers 100, a small computer system interface, or SCSI, bus 110A and a SCSI bus 110B connect the controllers 100A and 100B to a plurality of disks 120. For example, disks 120A, 120B, 120C and 120D are coupled to the SCSI bus 110A.

15 As stated above, the RAID system may be organized into physical representations of disks, called arrays, or logical representations of disks, called volumes. In one embodiment of the invention, the volumes may be repositories for information about the volume, while the arrays store no information. So, for example, information about a volume may be written on the volume itself. The array has no such storage capability.

20 In Figure 5, the RAID hardware devices 17a include a plurality of arrays 130. For example, disks 120C and disks 120D are joined to form array 130B. In one embodiment of the invention, each array 130 has at least one volume 140 associated with the array 130. In Figure 5, the array 130D includes two volumes 140E and 140F. However, array 130B includes only a single volume 140D.

25 The RAID hardware devices 17a may be presented in a number of different ways. For example, in Figure 5, the controllers 100A and 100B extend from the same secondary bus 98. Alternatively, these controllers 100 may extend from distinct buses.

From the physical RAID hardware devices 17a of Figure 5, a plurality of RAID objects 30a may be derived. In Figure 6, the RAID objects 30a include controller objects 100, bus objects 110, disk objects 120, array objects 130, and volume objects 140.

Each of the RAID objects 30a of Figure 6 represents a component of the RAID hardware devices 17a in Figure 5. For example, the RAID hardware devices 17a include eight disks 120A-120H. Thus, the RAID objects 30a include eight disk objects 120a-120h. Likewise, the RAID hardware devices 17a include two controllers 100A and 100B. Accordingly, in Figure 6, controller object 100a and 100b are two of the RAID objects 30a. The RAID objects 30a thus represent, using objects, the entire configuration of the RAID hardware devices 17a.

In one embodiment of the invention, each of the RAID objects 30a includes methods 32, properties 34 and events 36, which are consistent with the particular hardware features these objects 30a model. In Figure 7, for example, the controller object 100 includes a method 102, used to reset the controller of the RAID hardware devices 17a. Likewise, the controller object 100 includes properties 34, such as a bus counting method 103, used to report the number of buses on the controller, and a disk counting method 104, for reporting the number of disks on the controller. Each of the properties 34 is accessible to the RAID software 12a for communicating with the controller of the RAID hardware devices 17a.

In Figure 8, the bus object 110 includes a scan bus method 111 and several properties 34, including a bus indexing property 112, a bus identification property 113, a bus protocol property 114, a bus device count property 115 and a Boolean property 116, which tells whether the bus is presently scanning. As

with the controller object 100, the bus object 110 includes no events for the RAID software 12a to monitor. However, the RAID software 12a may set or retrieve any of the properties or may invoke the scan bus method 111, as desired.

5 In Figure 9, the disk object 120 includes methods 32, properties 34, and events 36 which allow the RAID software 12a to communicate with the plurality of disks, such as the eight disks 120A-120H of Figure 5. For example, a method 122, a method 125, and a method 126 may be used to mark a disk as normal, offline, and online, respectively. A total block count property 153 may be used
10 to retrieve the total block count of the disk.

 In addition to several methods 32 and properties 34, the disk object 120 includes events 36 which may permit the RAID software 12a to monitor the behavior of each disk modeled by the disk object 120. For example, a disk is normal event 161 may permit the RAID software 12a to be notified when the
15 disk has been marked as normal (following the invocation of the mark as normal method 122). Recall that the configuration library 20 provides two functions, event monitor on 68 and event monitor off 70, which allow the RAID software 12a to monitor one or more events 36 of the disk object 120.

 In Figures 10 and 11, the array object 130 and the volume object 140,
20 like the disk object 120, include methods 32, properties 34 and events 36. In one embodiment of the invention, the array is defined as a physical grouping of one or more disks. Accordingly, the array object 130 includes a method 131 to create an array and a method 132 to expand an array. A disk count property 134 enables the RAID software 12a to identify the number of disks 120 included
25 in the array 130. A volume degraded event 169 permits the RAID software 12a to receive notification when a volume 140 of the array 130 has degraded.

The volume object 140 (Figure 11) includes methods such as a method 141 and a method 142, for creating and deleting a volume 140, respectively, from the RAID hardware devices 17a. Properties such as number of bytes per block 171 and total number of blocks 172 provide other indicia about the volume 140. The volume object 140 further includes event notification for failed, migrating, and initializing volumes, an event 181, an event 182, and an event 183, respectively.

Using the above objects in conjunction with the functions of the configuration library 20, the RAID software 12a may perform a number of operations on the RAID hardware devices 17a. Simply by controlling the properties 34 and methods 32 of the RAID objects 30a, the RAID software 12a can configure the RAID hardware devices 17a.

As an example, the RAID software 12a may perform an operation to determine the devices which are located on a bus. Looking back to Figure 5, suppose the RAID software 12a wants to determine which devices are connected to the SCSI bus 110A of the system 14. An operation to scan the SCSI bus 110A according to one embodiment of the invention includes invoking the method invocation function 62 of the configuration library 20 (see Figure 3). In Figure 12, the method invocation function 62 is passed two parameters (block 202). The object 30 for which a method 32 is invoked is provided as a parameter. In the RAID objects 30a, the available objects are controller objects 100, bus objects 110, disk objects 120, array objects 130, and volume objects 140. To scan the SCSI bus 110A, the bus object 110a is passed as a parameter. The method 32 to invoke from the passed object is also provided as a parameter.

Looking back to Figure 8, the scan bus method 111 is the only method 32 available for the bus object 110, and is the desired method for performing a scan

operation upon the SCSI bus 110A. Accordingly, the method invocation function 62 is invoked, and the bus object 110 and the scan bus method 11 are passed as parameters (block 202).

The get property function 64 is next invoked (block 204). The get property function 64 is passed three parameters: the object for which a property is to be retrieved, the property to be retrieved, and a location for storing the property value. As with the method invocation function 62, the bus object 110 is passed as a parameter. Also, the bus is presently scanning property 116, and a memory location for storing the result, are passed as parameters. The property 116 returns a Boolean result.

Once the get property function 64 is complete, the allocated memory location contains a zero (false) or one (true) result. The next operation (diamond 206) tests the value of the property 116. If true, the property get function 64 is once again invoked (block 204).

If the property 116 is false, however, the get property function 64 is invoked, this time with the bus device count property 115 passed as a parameter (block 208). The bus device count property 115 indicates to the RAID software 12a how many devices were found on the SCSI bus 110A as a result of the scanning operation. Accordingly, an integer value is stored in the bus device count memory location once the property is retrieved by the get property function 64. The scanning operation is thus complete (block 210).

Thus, an interface to hardware includes a configuration library and objects to model the hardware. Software programs using the interface need not understand how to communicate with the hardware. Instead, the software programs may communicate with the interface. In turn, the interface communicates with the hardware. Such an organization allows the software to

be written even before the hardware is implemented, prior to the addition of new or different hardware, or under other conditions in which the hardware requirements are unknown.

5 While the present invention has been described with respect to a limited number of embodiments, those skilled in the art will appreciate numerous modifications and variations therefrom. It is intended that the appended claims cover all such modifications and variations as fall within the true spirit and scope of this present invention.

What is claimed is:

1 1. A method, comprising:
2 defining a plurality of hardware devices as a plurality of objects;
3 providing a plurality of tools to perform a plurality of operations on
4 the plurality of objects;
5 executing a software program to use the plurality of tools; and
6 responding to the plurality of operations by the plurality of
7 hardware devices.

1 2. The method of claim 1, wherein defining the plurality of hardware
2 devices as a plurality of objects further comprises:
3 assigning a plurality of properties to the plurality of hardware
4 devices; and
5 assigning a plurality of methods to the plurality of hardware
6 devices.

1 3. The method of claim 2, wherein defining the plurality of hardware
2 devices as a plurality of objects further comprises:
3 assigning a plurality of events to the plurality of hardware devices.

1 4. The method of claim 3, wherein providing a plurality of tools to
2 perform operations on the plurality of objects further comprises:
3 providing a function for invoking a method of an object;
4 providing a function for setting a property of an object; and
5 providing a function for retrieving a property of an object.

1 5. The method of claim 3, wherein providing a plurality of tools to
2 perform operations on the plurality of objects further comprises:
3 providing a function for monitoring an event of an object; and
4 providing a function for ending monitoring an event of an object.

1 6. An article comprising a medium storing instructions that cause a
2 processor-based system to:
3 receive a request from a software program;
4 act upon a plurality of objects based upon the request received;
5 and
6 manipulate a plurality of hardware devices modeled by the plurality
7 of objects.

1 7. The article of claim 6, further storing instructions that cause a
2 processor-based system to use a plurality of configuration library tools to act
3 upon a plurality of objects.

1 8. The article of claim 7, further storing instructions that cause a
2 processor-based system to invoke a plurality of methods of the plurality of
3 objects.

1 9. The article of claim 7, further storing instructions that cause a
2 processor-based system to retrieve a plurality of properties of the plurality of
3 objects.

1 10. The article of claim 7, further storing instructions that cause a
2 processor-based system to monitor a plurality of events for the plurality of
3 objects.

1 11. A system, comprising:
2 a processor;
3 a plurality of hardware devices; and
4 a medium including a software program which:
5 models the plurality of hardware devices as a plurality of
6 objects, wherein the plurality of objects comprise a plurality of methods and a
7 plurality of properties;
8 provides a plurality of tools for performing a plurality of
9 operations on the plurality of objects; and
10 invokes the plurality of hardware devices to respond to the
11 plurality of operations performed on the plurality of objects.

1 12. The system of claim 11, wherein the software program further
2 models the plurality of hardware devices as a plurality of events.

1 13. The system of claim 11, wherein the software program further
2 performs operations on the plurality of objects by invoking one of the plurality of
3 methods of one of the plurality of objects.

1 14. The system of claim 11, wherein the software program further
2 performs operations on the plurality of objects by setting one of the plurality of
3 properties of one of the plurality of objects.

1 15. The system of claim 12, wherein the software program further
2 performs operations on the plurality of objects by monitoring one of the plurality
3 of events of one of the plurality of objects.

1 16. A system, comprising:
2 a processor;
3 a plurality of disks; and
4 a memory storing software which:
5 models the plurality of disks as a plurality of disk objects;
6 provides a plurality of tools for performing a plurality of
7 operations on the plurality of disk objects; and
8 invokes a response by the plurality of disks to the plurality of
9 operations performed on the plurality of disk objects.

1 17. The system of claim 16, wherein the software program is stored in
2 the memory.

1 18. The system of claim 16, further comprising:
2 a plurality of buses; and
3 a plurality of controllers.

1 19. The system of claim 18, further comprising a memory storing
2 software which:

3 models the plurality of buses as a plurality of bus objects; and
4 models the plurality of controllers as a plurality of controller
5 objects.

1 20. The system of claim 19, further comprising a memory storing
2 software which:

3 models the plurality of volumes as a plurality of volume objects;
4 and
5 models the plurality of arrays as a plurality of array objects.

1 21. The system of claim 20, further comprising a memory storing
2 software which invokes a response to the plurality of operations by:

3 the plurality of buses for operations performed on the plurality of
4 bus objects; and
5 the plurality of controllers for operations performed on the plurality
6 of controller objects.

1 22. An object comprising:
2 a plurality of methods to model operations performed upon a
3 device;

4 a plurality of properties to model attributes of the device; and
5 a plurality of events to model actions of the device.

1 23. The object of claim 22, wherein the methods comprise parameters
2 of the object.

1 24. The object of claim 23, wherein the parameters comprise
2 properties of the object.

1 25. A system, comprising:
2 an interface, comprising:
3 a plurality of functions; and
4 a plurality of objects coupled to the plurality of functions;
5 and
6 a plurality of devices coupled to the interface, wherein a
7 software program may control the plurality of devices by communicating with the
8 interface.

1 26. The system of claim 25, wherein the plurality of functions further
2 comprises a function for retrieving a property of one of the plurality of objects.

1 27. The system of claim 26, wherein the function for retrieving a
2 property of one of the plurality of functions further comprises:
3 a parameter to identify the object for which the property is
4 retrieved;
5 a parameter to identify the property to be retrieved; and
6 a parameter for storing a result.

METHOD FOR MODELING HARDWARE USING SOFTWARE

Abstract of the Disclosure

5 An interface to one or more hardware devices includes a configuration library and objects to model the hardware. Software programs using the interface need not understand how to communicate with the hardware. Instead, the software programs may communicate with the interface. In turn, the interface communicates with the hardware. The software may be written when the hardware implementation features are unknown.

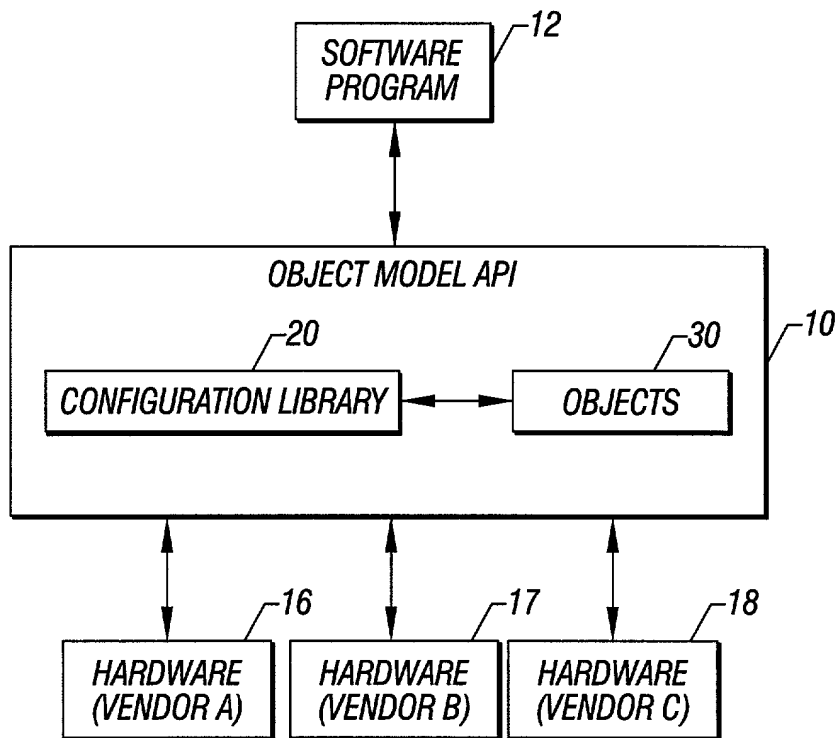


FIGURE 1

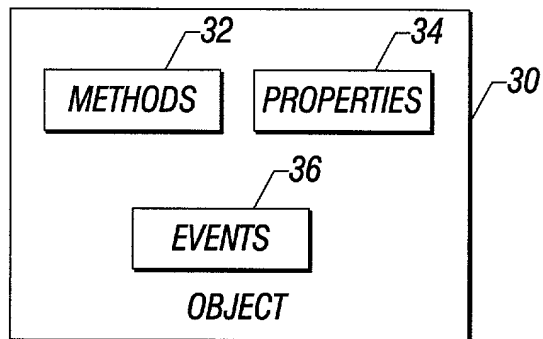


FIGURE 2

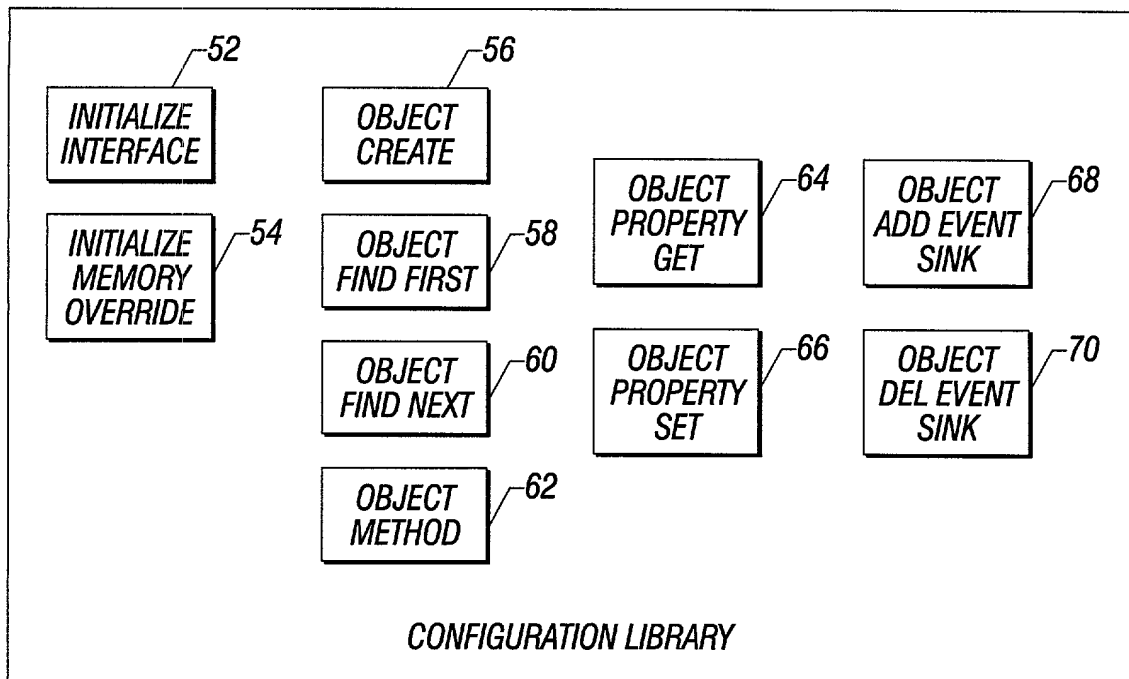


FIGURE 3

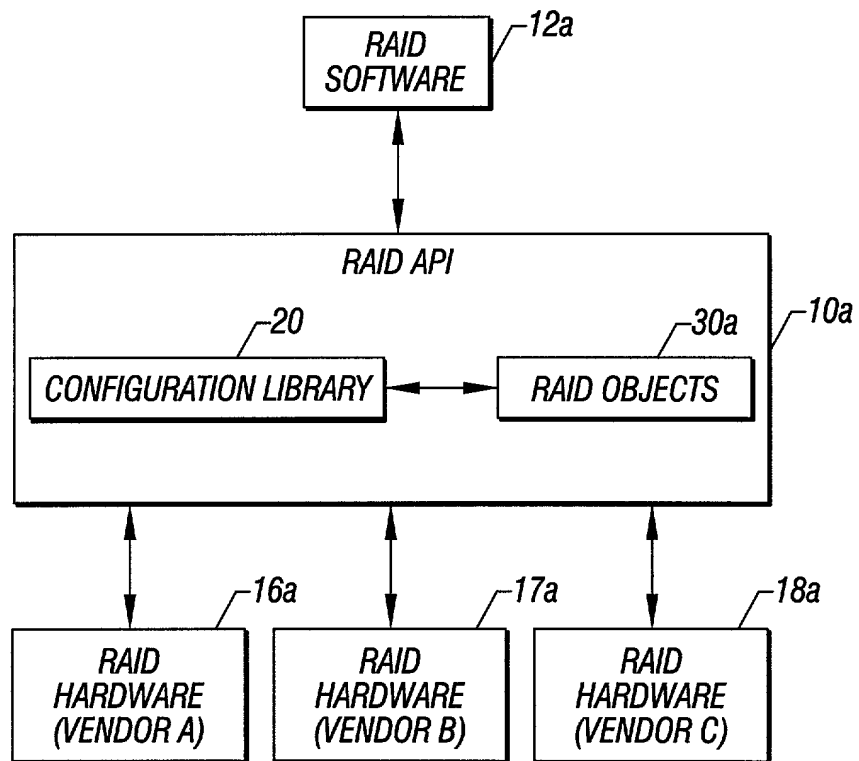


FIGURE 4

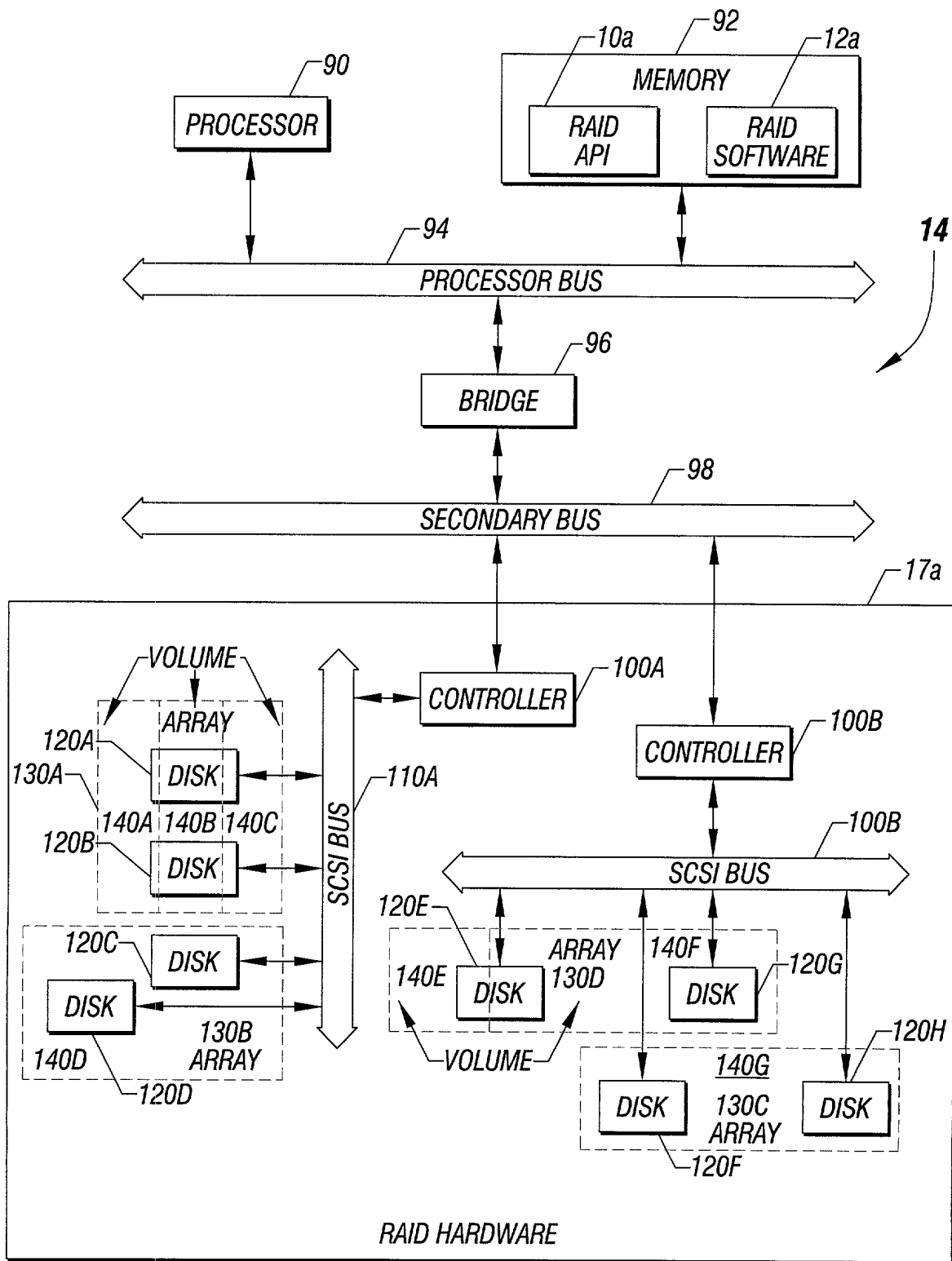
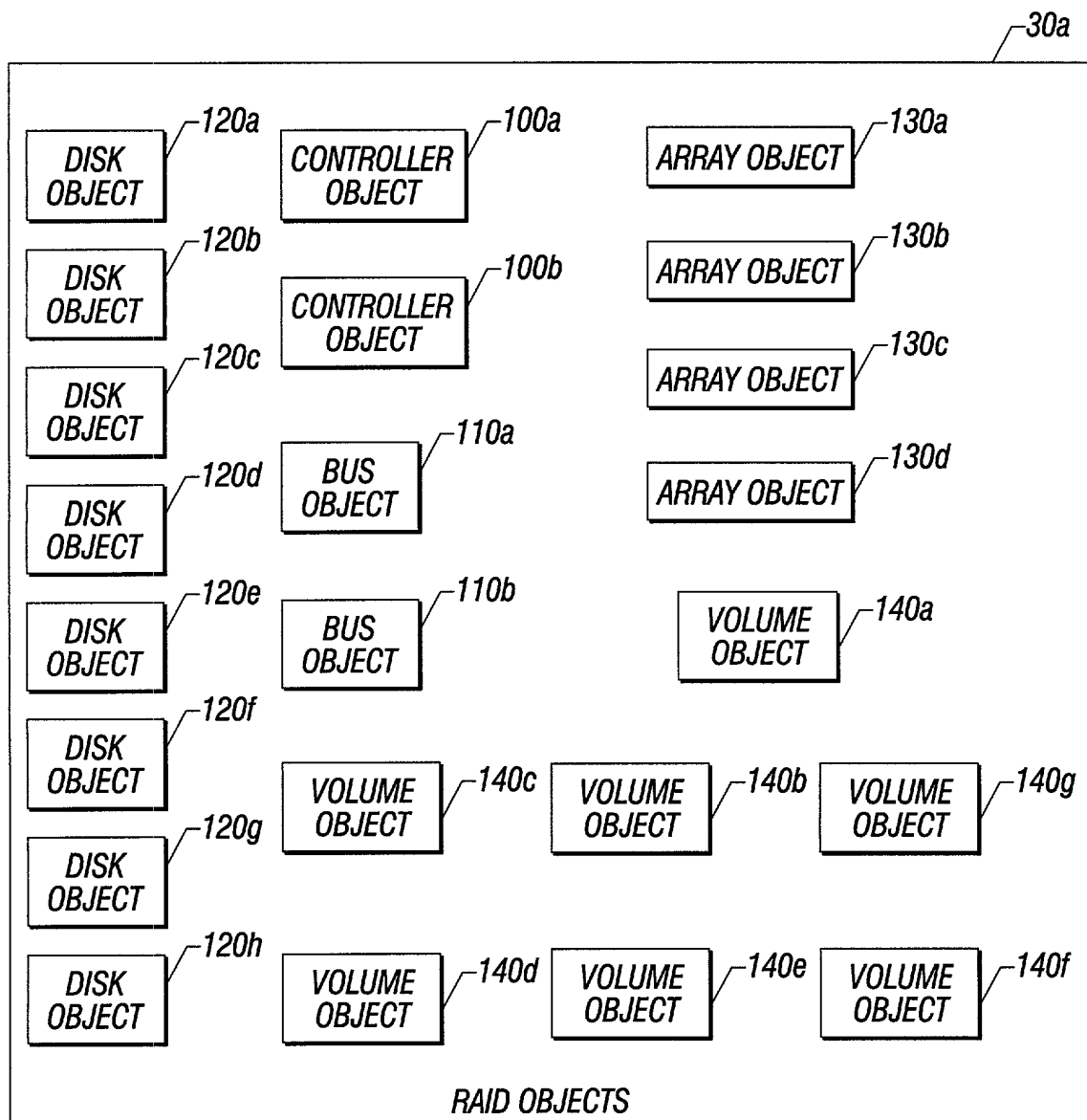


FIGURE 5



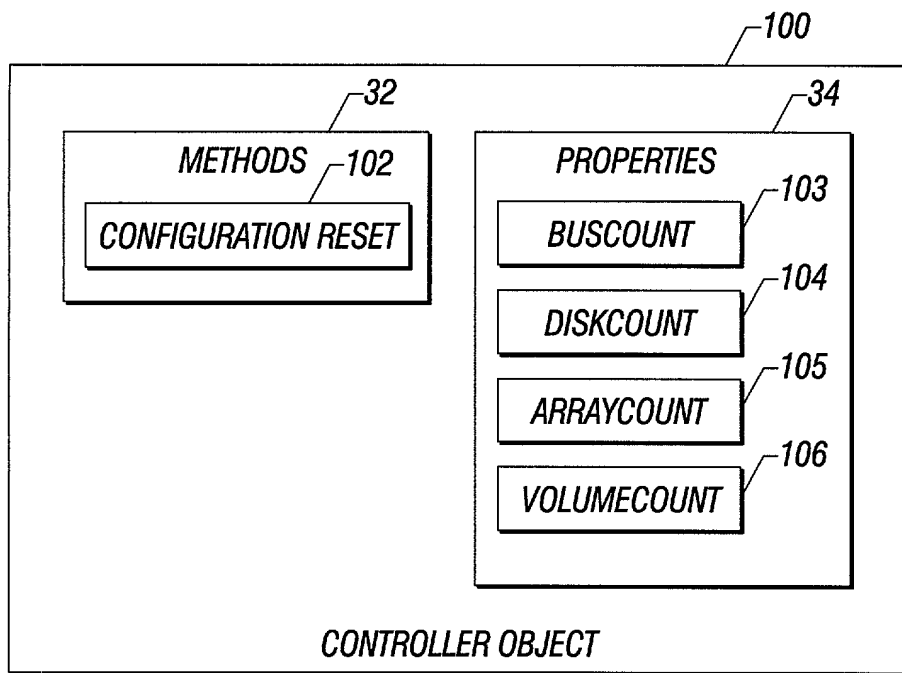


FIGURE 7

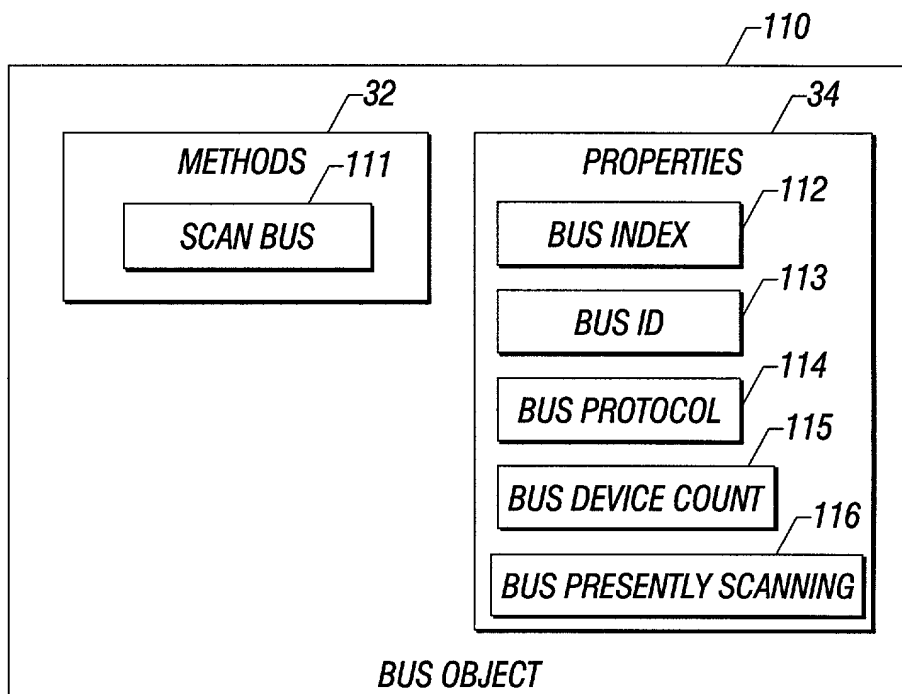


FIGURE 8

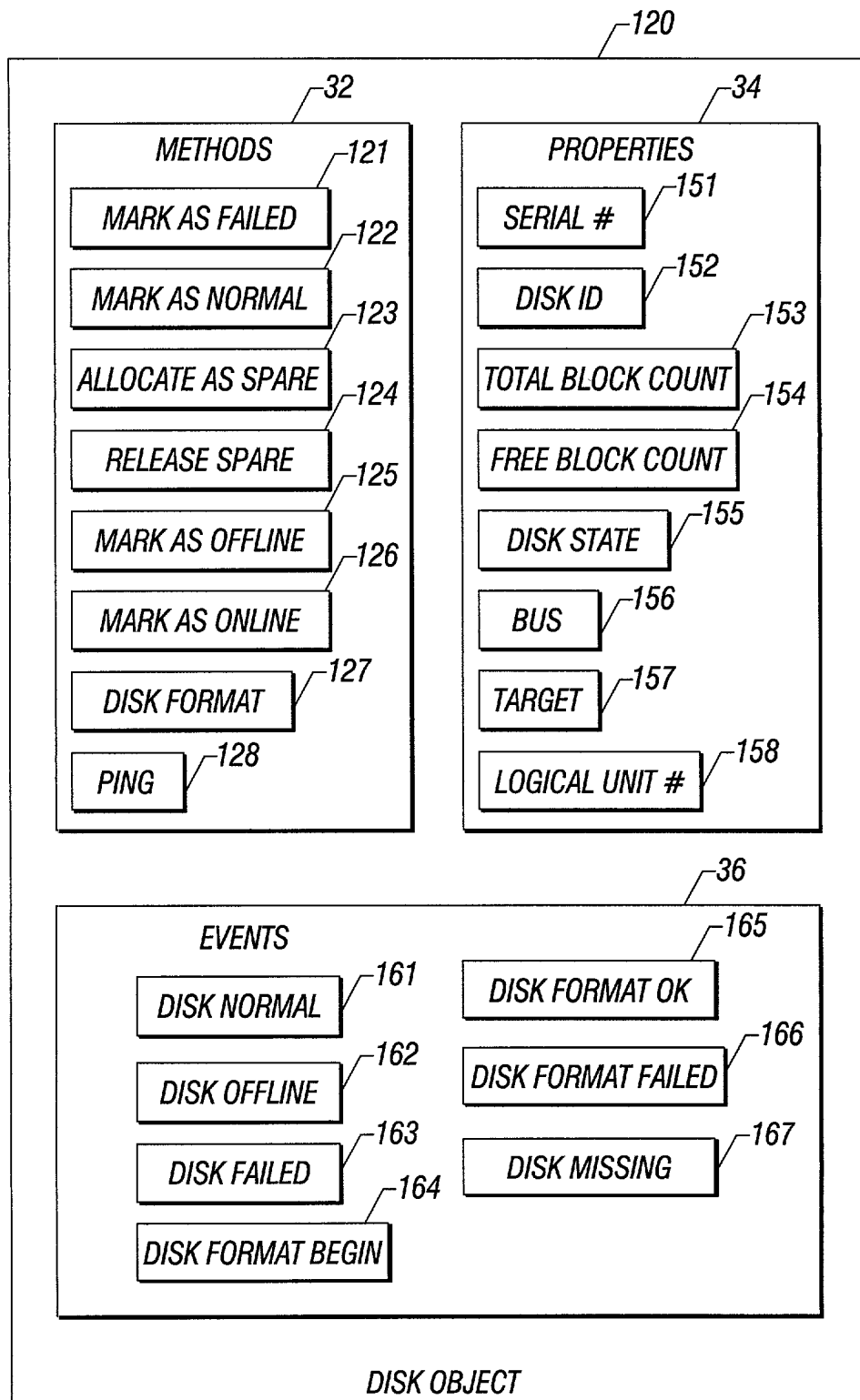


FIGURE 9

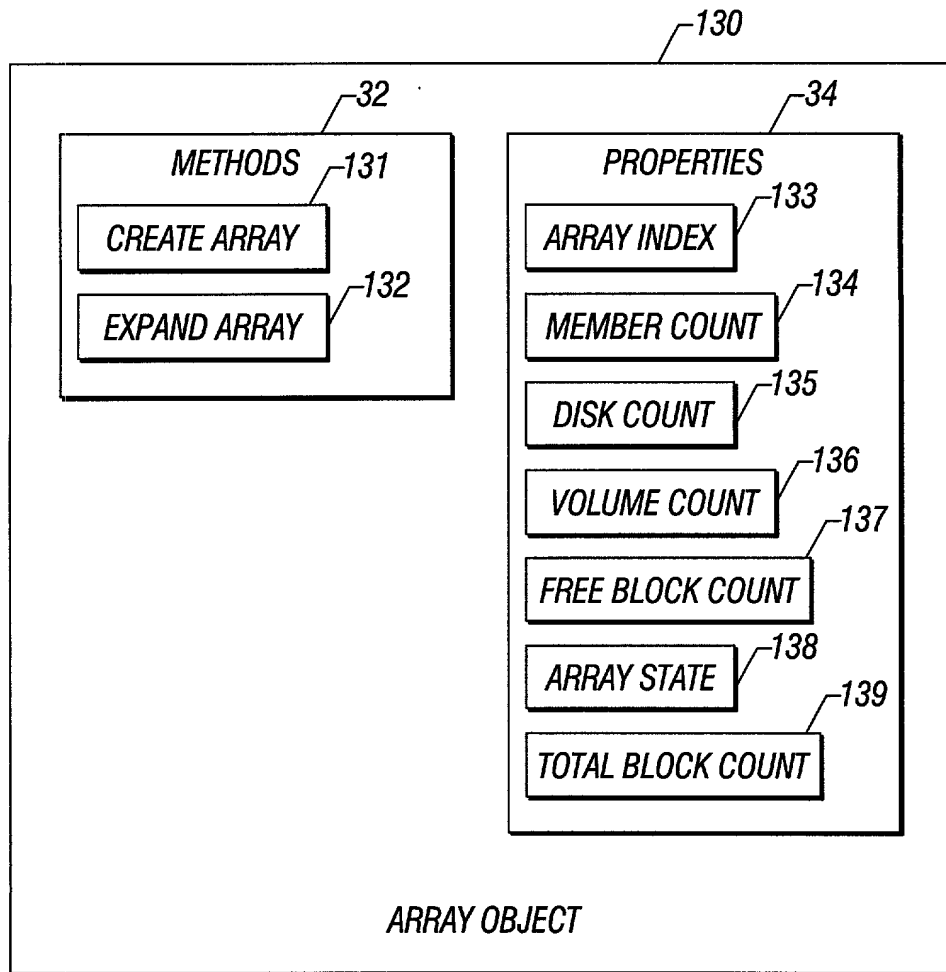


FIGURE 10

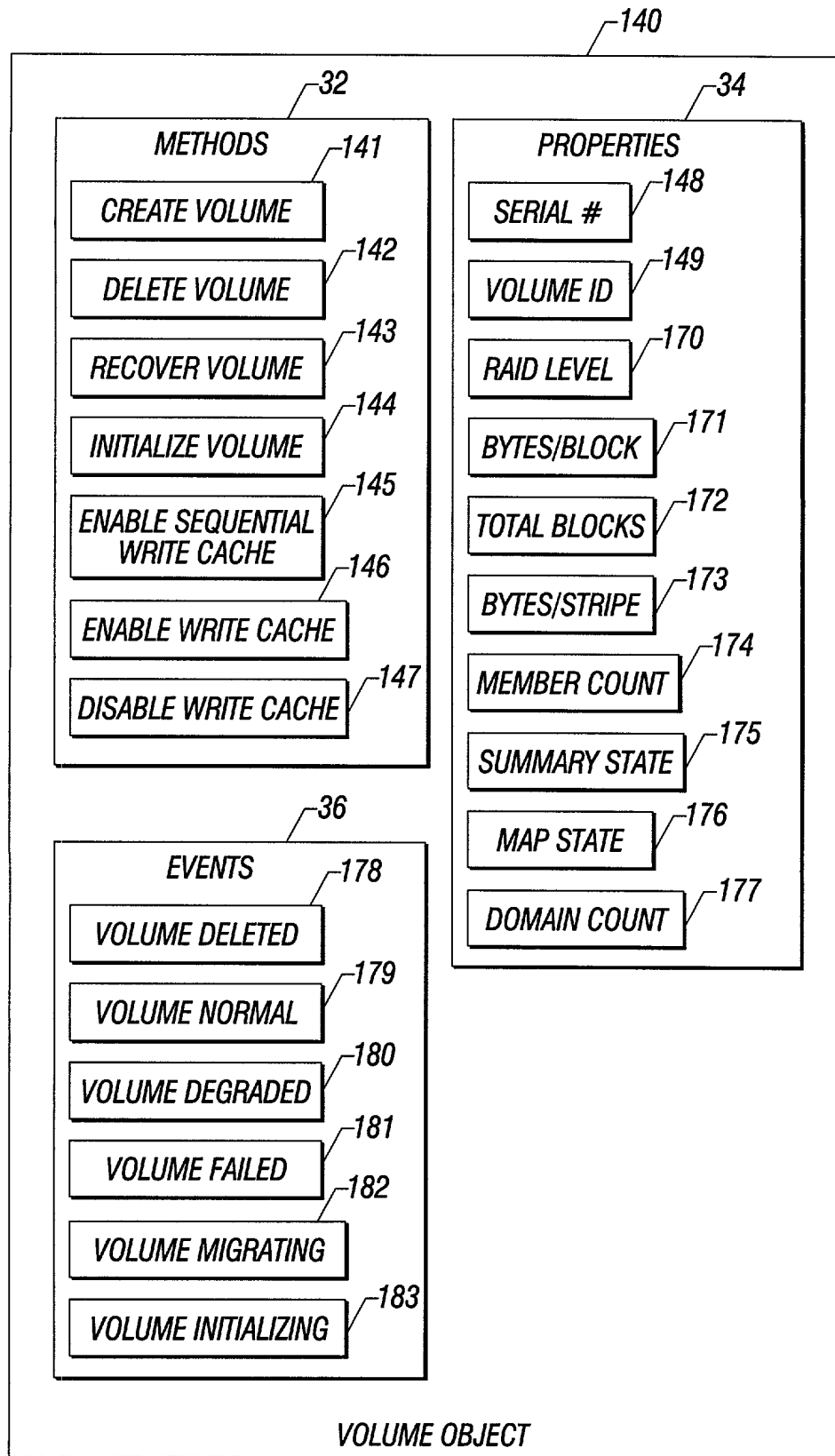


FIGURE 11

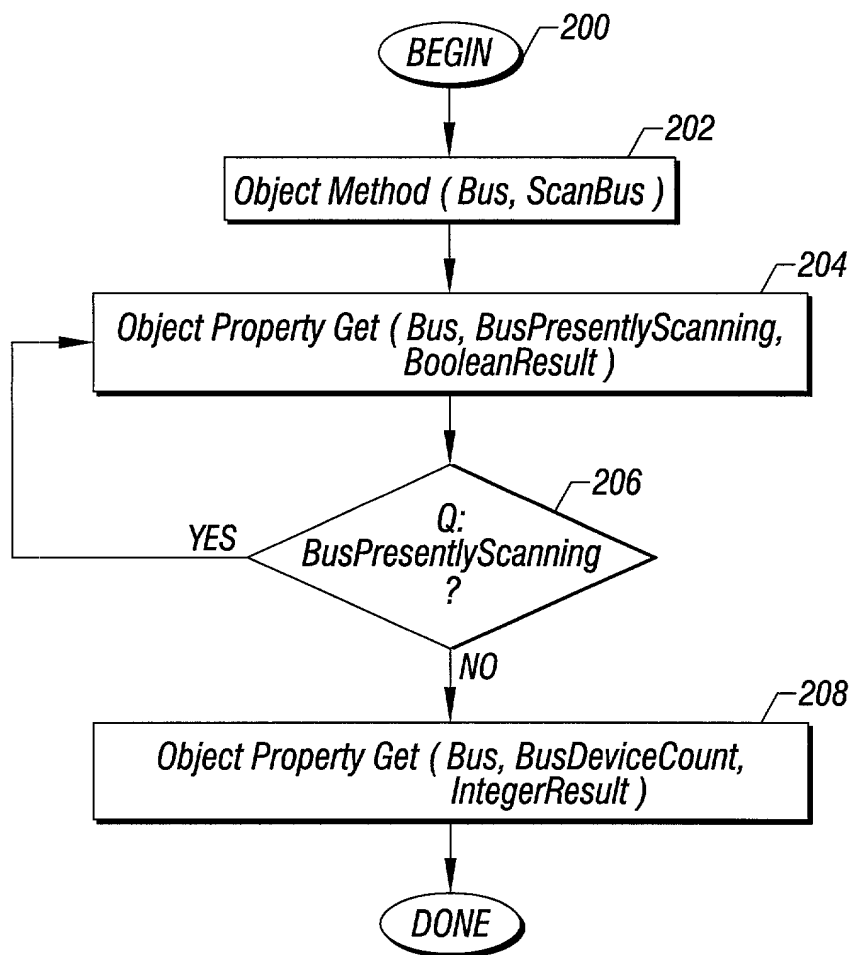


FIGURE 12

DECLARATION AND POWER OF ATTORNEY FOR PATENT APPLICATION

As a below named inventor, I hereby declare that:

My residence, post office address and citizenship are as stated below, next to my name.

I believe I am the original, first, and sole inventor (if only one name is listed below) or an original, first, and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled

METHOD FOR MODELING HARDWARE USING SOFTWARE

the specification of which

X	is attached hereto.
	was filed on _____ as
	United States Application Number _____
	or PCT International Application Number _____
	and was amended on _____
	(if applicable)

I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claim(s), as amended by any amendment referred to above. I do not know and do not believe that the claimed invention was ever known or used in the United States of America before my invention thereof, or patented or described in any printed publication in any country before my invention thereof or more than one year prior to this application, that the same was not in public use or on sale in the United States of America more than one year prior to this application, and that the invention has not been patented or made the subject of an inventor's certificate Issued before the date of this application in any country foreign to the United States of America on an application filed by me or my legal representatives or assigns more than twelve months (for a utility patent application) or six months (for a design patent application) prior to this application.

I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal Regulations, Section 1.56.

I hereby claim foreign priority benefits under Title 35, United States Code, Section 119(a)-(d), of any foreign application(s) for patent or inventor's certificate listed below and have also identified below any foreign application for patent or inventor's certificate having a filing date before that of the application on which priority is claimed:

Prior Foreign Application(s):			Priority Claimed	
Number	(Country)	(Day/Month/Year Filed)	Yes	No
Number	(Country)	(Day/Month/Year Filed)	Yes	No
Number	(Country)	(Day/Month/Year Filed)	Yes	No

I hereby claim the benefit under title 35, United States Code, Section 119(e) of the United States provisional application(s) listed below:

_____	_____
(Application Number)	(Filing Date)
_____	_____
(Application Number)	(Filing Date)


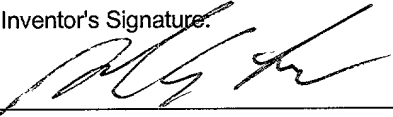
I hereby claim the benefit under Title 35, United States Code, Section 120 of any United States application(s) listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior United States application in the manner provided by the first paragraph of Title 35, United States Code, Section 112, I acknowledge the duty to disclose all information known to me to be material to patentability as defined in Title 37, Code of Federal regulations, Section 1.56 which became available between the filing date of the prior application and the national or PCT International filing date of this application:

_____	_____	_____
(Application Number)	Filing Date	(Status-patented, pending, abandoned)
_____	_____	_____
(Application Number)	Filing Date	(Status-patented, pending, abandoned)

I hereby appoint Timothy N. Trop, Reg. No. 28,994; Fred G. Pruner, Jr., Reg. No. 40,779, Dan C. Hu, Reg. No. 40,025; Coe F. Miles, Reg. No. 38,559, and John R. Merkling, Reg. No. 31,716 my patent attorneys, of TROP, PRUNER, HU & MILES, P.C., with offices located at 8554 Katy Freeway, Ste. 100, Houston, TX 77024, telephone (713) 468-8880, and Joseph R. Bond, Reg. No. 36,458; Richard C. Calderwood, Reg. No. 35,468; Sean Fitzgerald, Reg. No. 32,027; David J. Kaplan, Reg. No. 41,105; Leo V. Novakoski, Reg. No. 37,198; Naomi Obinata, Reg. No. 39,320; Thomas C. Reynolds, Reg. No. 32,488; Steven P. Skabrat, Reg. No. 36,279; Howard A. Skaist, Reg. No. 36,008; Steven C. Stewart, Reg. No. 33,555; Raymond J. Werner, Reg. No. 34,752; and Charles K. Young, Reg. No. 39,425; my patent attorneys, of INTEL CORPORATION; with full power of substitution and revocation, to prosecute this application and to transact all business in the Patent and Trademark Office connected herewith.

Send correspondence to Timothy N. Trop, TROP, PRUNER, HU & MILES, P.C., 8554 Katy Freeway, Ste. 100, Houston, TX 77024 and direct telephone calls to Timothy N. Trop, (713) 468-8880.

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

Full Name of Sole/First Inventor: STEVEN C. DAKE	
Inventor's Signature: 	Date: 12/20/99
Residence: TEMPE, AZ	Citizenship: UNITED STATES
Post Office Address: 985 E. BRENTUP, TEMPE, AZ 85823	
Full Name of Second/Joint Inventor: PAUL E. LUSE	
Inventor's Signature: 	Date: 12/20/99
Residence: CHANDLER, AZ	Citizenship: UNITED STATES
Post Office Address: 979 W. MYRTLE DRIVE, CHANDLER, AZ 85248	

INTL-0278 -US (P7627)